Les modèles créateurs

- Ces modèles englobent tous les patterns dont le rôle est de créer des objets à la demande. Nous étudierons les patterns :

  - **Singleton**
  - **Fabrication**
  - **Fabrique abstraite**
  - **Monteur**

Le pattern créateur Singleton

- **La motivation**
  - La quasi-totalité des logiciels communique lors de leur exécution avec une base de données.
  - Pour gérer une seule et même transaction lors d'une exécution d'un logiciel, il faut être assuré qu' à tout instant de l'exécution on manipule la même session (au sens base de données).

Le pattern créateur Singleton

- **L'intention de ce pattern**
  - Ce pattern permet pour une classe donnée de ne fabriquer qu'un seul objet et donc de manipuler à tout instant d'une exécution le même objet.

- **Les constituants de ce pattern**
  - Il n'y a qu'un seul constituant : la classe qui fournit l'unique objet.

Le pattern créateur Singleton (2)

- **La collaboration**
  - Cette classe collabore avec les autres classes à l'aide :
    - D'une méthode statique qui permet de récupérer une référence sur l'unique objet de la classe
    - De l'ensemble des méthodes qui définissent le comportement de cet objet.

Le pattern créateur Fabrique Abstraite

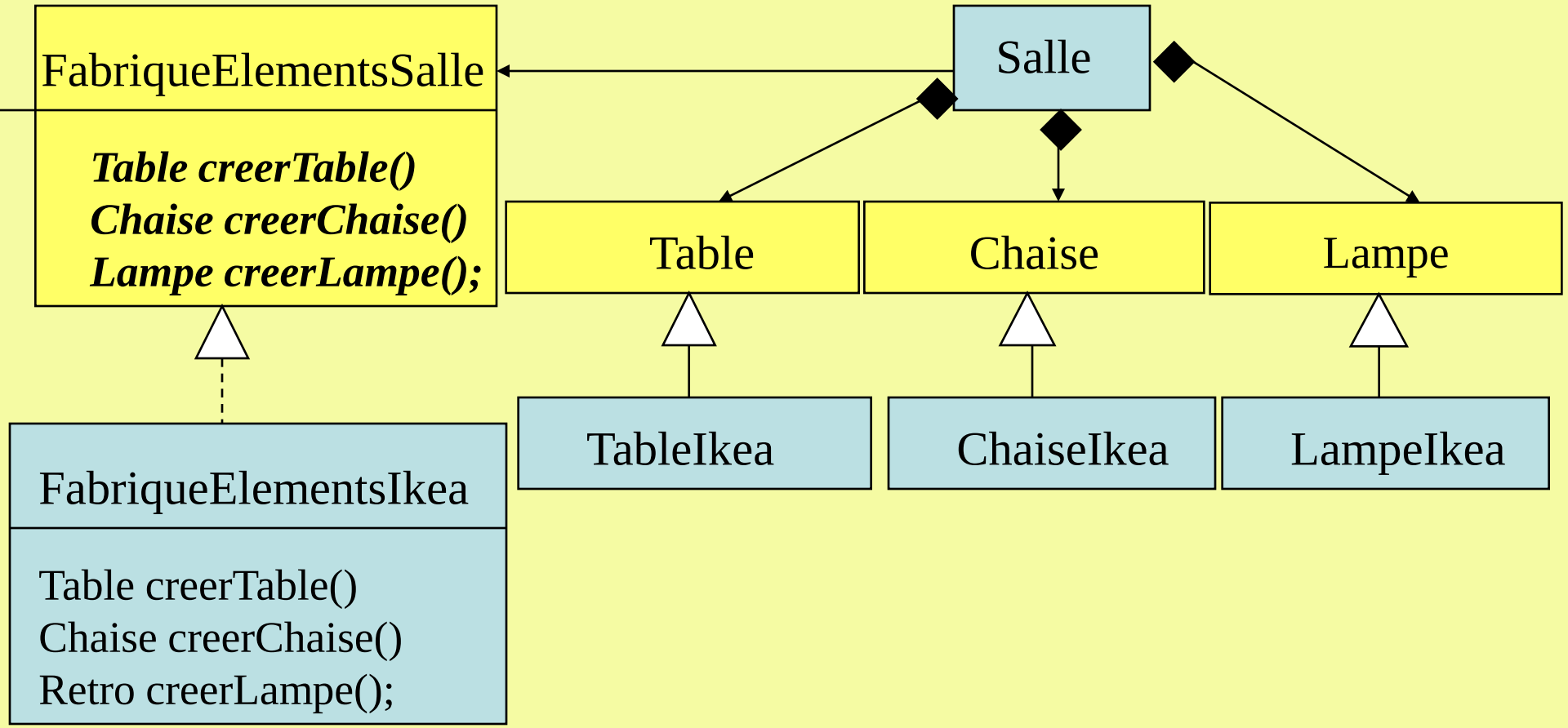
- **La motivation**
  - On désire définir à l'aide d'un logiciel, le contenu d'une salle de travail : tables, chaises et des lampes
  - D'une salle à l'autre ou d'une école à l'autre, les types des éléments peuvent être différents
  - Pour une salle donnée, il ne faut donc pas « coder en dur » les différents éléments de cette dernière. Cela rendrait difficile un changement ultérieur du type des éléments

Le pattern créateur Fabrique Abstraite

- **L'intention**
  - La fabrique abstraite fournit une interface pour la création d'une famille d'objets liés entre eux ou apparentés sans qu'il soit nécessaire de spécifier le type des différents objets

Le pattern créateur Fabrique Abstraite

- **Retour sur notre motivation**

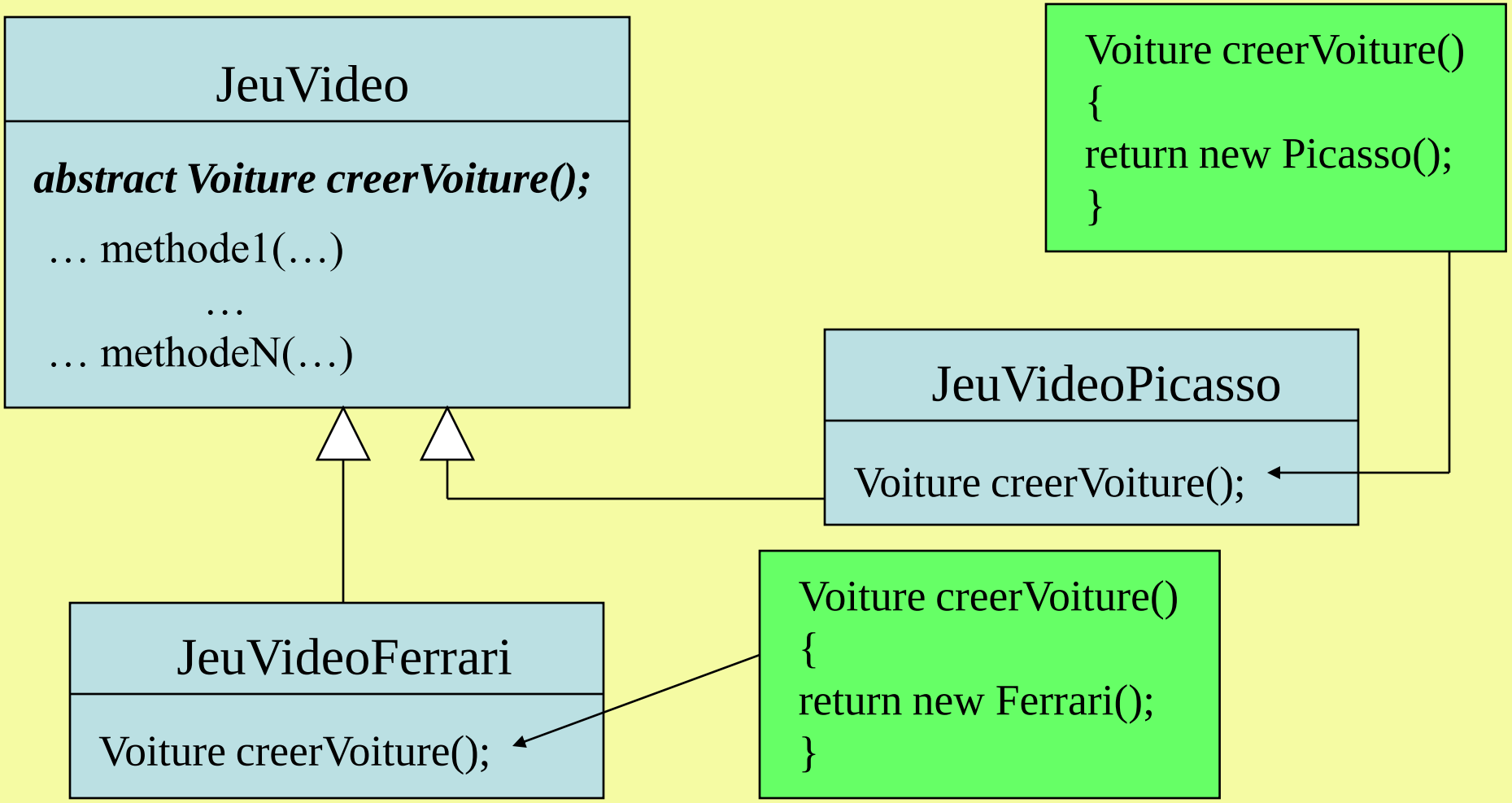


Le pattern créateur Fabrication

- **La motivation**
  - On conçoit un jeu vidéo qui permet de piloter des voitures. On veut concevoir de nouvelles voitures indépendamment de la conception du jeu vidéo.
  - Le jeu vidéo connaît donc l'interface d'une voiture mais ne connaît pas le type des voitures utilisées.
  - Pour disposer d'une voiture, le jeu vidéo doit pouvoir fabriquer la voiture sans en connaître le type !!!.

Le pattern créateur Fabrication

- **Retour sur notre motivation**



Le pattern créateur Fabrication (2)

- **L'intention**
  - Définir une classe (abstraite) qui implémente toutes ses méthodes qui utilisent des objets et ne fait que déclarer les méthodes de fabrication des objets utilisés.
  - La méthode de fabrication permet à la classe abstraite de déléguer l'instanciation des objets utilisés à d'autres classes.

Le pattern créateur Fabrication

- **Les constituants**
  - Produit : l'interface des objets créés par la fabrication
  - ProduitConcret : Une classe qui implémente l'interface Produit.
  - Facteur : La classe qui déclare la méthode de fabrication et qui implémente les autres méthodes
  - FacteurConcret : La classe qui hérite de la classe Facteur en surchargeant la méthode de fabrication

Le pattern des patterns créateurs (1)

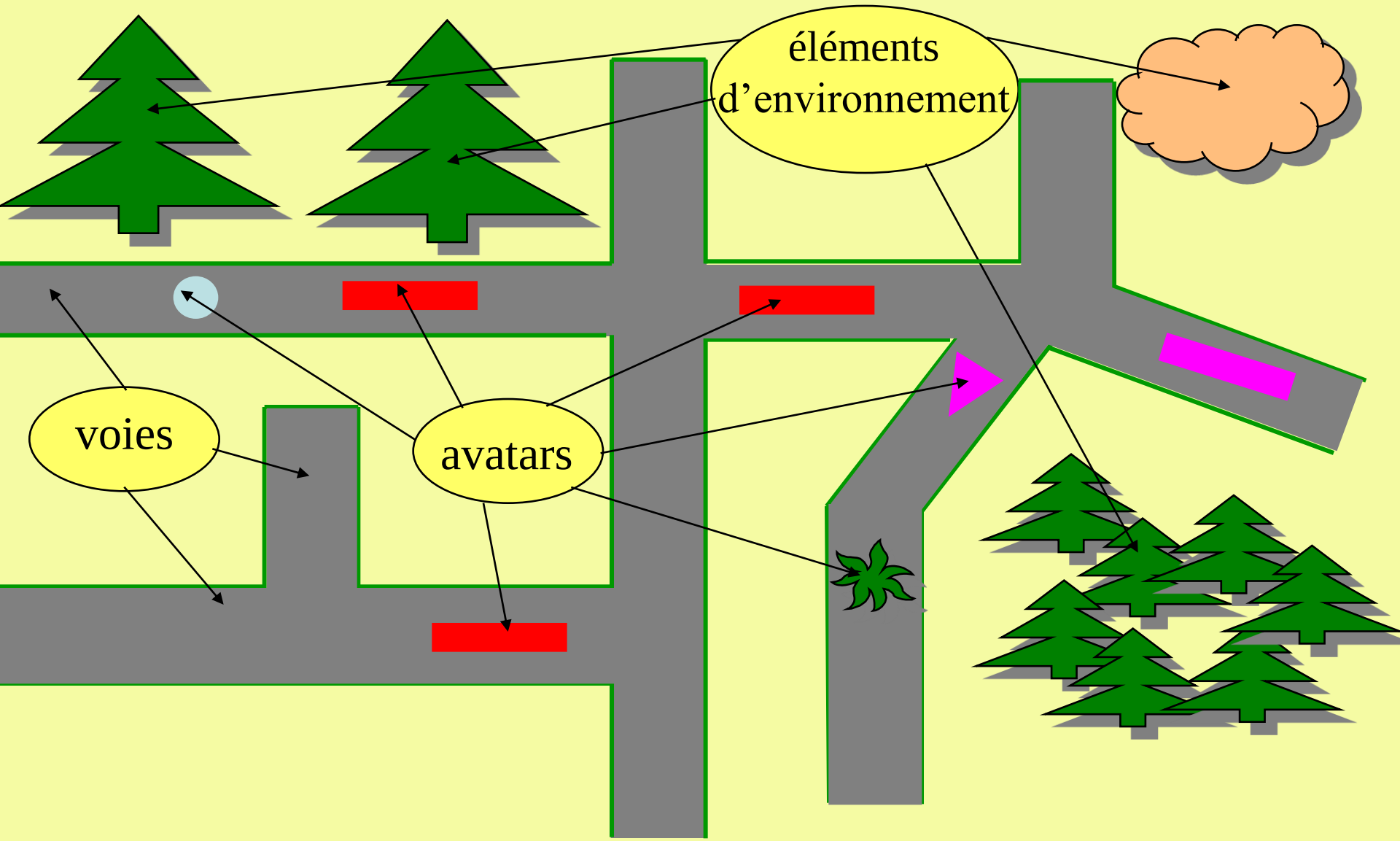
- Exemple :
  - Un petit logiciel d'animation de formes vivantes qui circulent sur des voies.
  - Ces chemins se trouvent dans un espace. Un espace a en plus de ses voies des éléments d'environnement.

Le pattern des patterns créateurs (1)

- On peut créer des espaces formés
  - de portions de voies
  - de croisement de voies
  - d'un environnement (arbre, rivière, …)

- A tout moment, on peut créer de nouveaux avatars et les poser sur une voie d'un espace.

Le pattern des patterns créateurs



Le pattern des patterns créateurs

- La classe Animation utilise comme objets
  - des espaces et leurs éléments d'environnement
  - des portions de voies
  - des croisements
  - des avatars

Le pattern des patterns créateurs

1. On va donc créer une fabrique abstraite

interface IFabriqueAnimation

ElementEnvironnement
creerElementEnvironnement(…)

Espace creerEspace(…)

Croisement creerCroisement(…)

PortionVoie creerPortionVoie(…)

Avatar creerAvatar(…)

Le pattern des patterns créateurs

2. Pour chaque objet manipulé (espace, élément d'environnement, avatar, voie, croisement), on va définir son interface d'utilisation dans la classe Animation
   - IEspace
   - IElementEnvironnement
   - IVoie
   - ICroisement
   - IAvatar

Le pattern des patterns créateurs

Ces deux premiers points permettent de concevoir la classe Animation en ne connaissant pas le véritable type des objets qu'elles utilisent

Le pattern des patterns créateurs

- Pour les deux points suivants, on utilise le pattern Fabrication

3. La classe Animation sera une classe abstraite qui
   - déclarera une méthode abstraite qui retournera une fabrique via une référence de IFabriqueAnimation
   - implémentera toutes les méthodes du logiciel qui utilisent, à travers leur interface, les objets fabriqués.

4. Enfin, on crée une classe concrète AnimationConcrete qui dérive d'Animation en surchargeant la méthode qui crée une fabrique

Le pattern des patterns créateurs

```
/**
Cette classe implémente tout le logiciel a l'exception du mécanisme d'instanciation des objets délégués à une sous classe
*/
class Animation {
        private IEspace esp;

        public IFabricationAnimation creerFabricationAnimation();

        void run(…) {
        esp = creerFabricationAnimation().creerEspace();
        …
        IAvatar av = creerFabricationAnimation().creerAvatar(…);
        esp.ajouterAvatar(av);
        …
        methodei(…);
        }

        void methodei(…) { esp.ajouterVoie(creerFabricationAnimation().creerVoie(…));
        …
        };}
```

Le pattern des patterns créateurs

```
// Une classe concrète d'animation
class AnimationRurale extends Animation {

    IFabricationAnimation creerFabricationAnimation(…) {
       return new FabricationAnimationRurale.getInstance(…);
    }

    public static void main (String [] args) {
       AnimationRurale ar = new AnimationRurale(…);


    ar.run(…);
    }
}
```

Le pattern des patterns créateurs

5. La classe FabricationAnimationRurale peut être implémentée en Singleton afin de contrôler à l'aide d'un unique objet l'ensemble des objets fabriqués

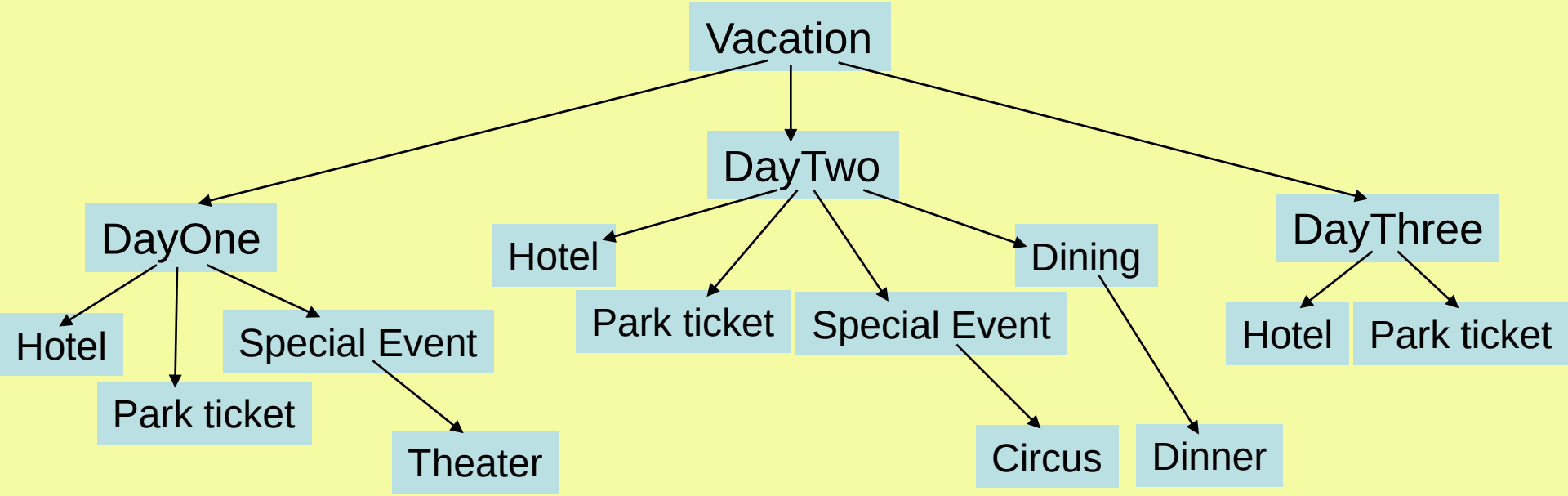
Le pattern des patterns créateurs

En résumé dans toute application logicielle, on peut systématiquement utiliser

- Le pattern Fabrique Abstraite pour offrir à la classe logiciel une interface de fabrication des objets utilisés par le logiciel
- Le pattern Fabrication pour implémenter la fabrication de la fabrique d'objets. Cette fabrication masque le type de la fabrique d'objets à la classe logicielle
- Le pattern Singleton pour garantir l'utilisation de la même fabrique dans les actions du logiciel

Le pattern créateur Monteur

- Motivation
  - Système de préparation de vacances d'un parc d'attraction



Structure de données complexe :
Besoin de flexibilité

Le pattern créateur Monteur (2)

- Intention
  - Simplifier la création d'objets complexes en définissant un classe dont le but est de construire les instances d'une autre classe
  - Le monteur produit un produit principal qui peut être composé de plusieurs classes mais il y a toujours une classe principale

Le pattern créateur Monteur (3)

- Constituants
  - Directeur : appelle les méthodes de création sur le Monteur pour créer les différentes parties du Produit et le Produit lui-même
  - MonteurAbstrait : définit les méthodes pour créer le Produit et ses parties
  - MonteurConcret : réalise les méthodes pour créer le Produit et ses parties
  - Produit : l'objet à monter

Le pattern créateur Monteur (4)